

EE31806180845

DOCKET NUMBER: AT9-98-955

DATE: 4-29-99

Box PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

JC511 U.S. PTO

09/303356



04/29/99

Sir:

Transmitted herewith for filing is the Patent Application of:

Inventors: **David W. Bachmann and Terry L. Dunkle**
For: **METHOD FOR DEFERRED DELETION OF ENTRIES FOR A DIRECTORY SERVICE
BACKING STORE**

Enclosed are:

- ☒ Patent Specification and Declaration
- ☒ 4 sheets of drawing(s). (Informal)
- ☒ An assignment of the invention to International Business Machines Corporation (includes Recordation Form Cover Sheet).
- ☐ A certified copy of a ____ application.
- ☐ Information Disclosure Statement, PTO 1449 and copies of references.

The filing fee has been calculated as shown below:

| For | Number Filed | Number Extra | Rate | Fee |
|-----------------------------------|--------------|--------------|--------------|------------|
| Basic Fee | | | | 760 |
| Total Claims | 20-20 | 0 | x \$18 = | 0 |
| Indep. Claims | 5-3 | 2 | x \$78 = | 156 |
| Multiple Dep. Claims Presented | | | x \$260 = | 0 |
| | | | TOTAL | 916 |

- ☒ Please charge my Deposit Account No. 09-0447 in the amount of \$916.00. A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account 09-0447. A duplicate copy of this sheet is enclosed.
 - ☒ Any additional filing fees required under 37 CFR 1.16.
 - ☒ Any patent application processing fees under 37 CFR 1.17.

Respectfully submitted,

By:

Jeffrey S. LaBaw
Jeffrey S. LaBaw
Registration No. 31,633
Intellectual Property Law Dept.
IBM Corporation
11400 Burnet Road, Zip 4054
Austin, Texas 78758
Telephone (512) 823-0494

**METHOD FOR DEFERRED DELETION OF ENTRIES
FOR A DIRECTORY SERVICE BACKING STORE**

This application includes subject matter protected
5 by copyright. All rights are reserved.

BACKGROUND OF THE INVENTION

Technical Field

This invention relates generally to providing
directory services in a distributed computing
environment.

11 Description of the Related Art

A directory service is the central point where
network services, security services and applications
can form an integrated distributed computing
environment. Typical uses of a directory services may
16 be classified into several categories. A "naming
service" (e.g., DNS and DCE Cell Directory Service
(CDS)) uses the directory as a source to locate an
Internet host address or the location of a given
server. A "user registry" (e.g., Novell NDS) stores
21 information about users in a system composed of a
number of interconnected machines. The central
repository of user information enables a system
administrator to administer the distributed system as a
single system image. Still another directory service
26 is a "white pages" lookup provided by some e-mail
clients, e.g., Netscape Communicator, Lotus Notes,
Endora and the like).

With more and more applications and system services demanding a central information repository, the next generation directory service will need to provide system administrators with a data repository that can significantly ease administrative burdens. In addition, the future directory service must also provide end users with a rich information data warehouse that allows them to access department or company employee data, as well as resource information, such as name and location of printers, copy machines, and other environment resources. In the Internet/intranet environment, it will be required to provide user access to such information in a secure manner.

To this end, the Lightweight Directory Access Protocol (LDAP) has emerged as an IETF open standard to provide directory services to applications ranging from e-mail systems to distributed system management tools. LDAP is an evolving protocol that is based on a client-server model in which a client makes a TCP/IP connection to an LDAP server, sends requests, and receives responses. The LDAP information model in particular is based on an "entry," which contains information about some object. Entries are typically organized in a specified tree structure, and each entry is composed of attributes.

LDAP provides a number of known functions including query (search and compare), update, authentication and others. The search and compare operations are used to retrieve information from the database. For the search function, the criteria of the search is specified in a search filter. The search filter typically is a Boolean expression that consists of qualifiers including attribute name, attribute value and Boolean operators like AND, OR and NOT. Users can use the filter to perform complex search operations. One filter syntax is defined in RFC 2254.

LDAP thus provides the capability for directory information to be efficiently queried or updated. It offers a rich set of searching capabilities with which users can put together complex queries to get desired information from a backing store. Increasingly, it has become desirable to use a relational database for storing LDAP directory data. Representative database implementations include DB/2, Oracle, Sybase, Informix and the like. As is well known, Structured Query Language (SQL) is the standard language used to access such databases.

In implementing an LDAP directory service with a relational database backing store, deleting an entry from the directory involves deleting rows from several different tables. In particular, in addition to the LDAP entry table, which stores an entry ID, parent ID,

create and last modified times, together with the complete entry in string format, the schema includes a separate table for each attribute. When an entry is to be deleted, a global lock is placed on all of these

5 tables (including the entry table and its associated attribute tables) until the delete is processed. As a result, all other query activity into the database is locked out for whatever time period is required for the backing store to return an indication that the delete

10 operation has been completed. This is a very time consuming and computationally-intensive process.

BRIEF SUMMARY OF THE INVENTION

It is a primary object of this invention to reduce the time required to perform a delete operation in a directory service having a relational database backing store.

18 It is another object of the present invention to delete an entry from a directory without having to lock out all other query activity during the operation as is presently required by the prior art.

23 A further object of the invention is to reduce the apparent processing time required to delete an entry from a directory by deferring the actual deletion until execution of a cleanup handler thread.

It is thus object of the present invention to provide a simple and efficient technique for speeding up entry deletion by deferring the actual deletion of rows from a database, preferably until invocation of a cleanup routine.

28 A specific object of this invention is to provide a more efficient LDAP directory service having a relational database management system (DBMS) as a backing store.

33 A general object of this invention is to provide a reliable and scaleable enterprise directory solution, wherein a preferred implementation is LDAP using a DB/2 backing store.

The present invention overcomes the deficiencies of the prior art. When an entry is to be deleted, its entry in an entry table (e.g., the ldap_entry table) is tagged deleted, preferably by setting its creation time to a given value (e.g., a null value). This operation involves a change to only a single, unindexed field in a single row in a single table and, as a result, is quite efficient. At periodic intervals, a cleanup thread performs actual row deletions for any entry tagged as deleted. When searches are done in the directory, the invention preferably modifies the SQL query to exclude rows with a null change time, thus preventing deleted entries from being returned by the search.

In a preferred embodiment, a method for deleting entries from a directory in which directory information is stored in a set of database tables begins upon a request to delete a directory entry. In response, the directory entry is tagged, preferably by setting the entry's creation time to a null value. If a search query is received thereafter, the method excludes tagged entries from search results that would otherwise satisfy the search query. At a periodic interval, the routine searches for tagged entries, and references to the tagged entries are then deleted throughout the set of database tables. Thus, the inventive method defers entry deletions to enable directory queries to be

processed even if deleted entries have not yet been
fully expunged from the directory.

The foregoing has outlined some of the more
pertinent objects and features of the present

5 invention. These objects and features should be
construed to be merely illustrative of some of the more
prominent features and applications of the invention.
Many other beneficial results can be attained by
applying the disclosed invention in a different manner
10 or modifying the invention as will be described.

Accordingly, other objects and a fuller understanding
of the invention may be had by referring to the
following Detailed Description of the preferred
embodiment.

15

006372.00203:0423162.01

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in connection with the accompanying drawings in which:

21 **Figure 1** is a representative LDAP directory service implementation;

Figure 2 is a simplified LDAP directory;

Figure 3 is a flowchart of an LDAP directory session;

26 **Figures 4A-4B** show representative LDAP directory service implementations having relational database backing store;

Figure 5 is a representative LDAP entry table prior to being modified in accordance with the present
31 invention;

Figure 6 is the LDAP entry table of Figure 5 after being modified to mark a given entry as deleted in accordance with the invention; and

Figure 7 is a simplified flowchart of the
36 inventive cleanup handler routine for deleting tagged entries in the entry and attribute tables.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A block diagram of a representative LDAP directory service in which the present invention may be implemented is shown in **Figure 1**. As is well-known, LDAP is the lightweight directory access protocol, and this protocol has been implemented in the prior art, e.g., as either a front end to the X.500 directory service, or as a standalone directory service. According to the protocol, a client machine makes a TCP/IP connection to an LDAP server **12**, sends requests and receives responses. LDAP server **12** supports a directory **21** as illustrated in a simplified form in **Figure 2**. Each of the client and server machines further include a directory runtime component **25** for implementing the directory service operations as will be described below.

The directory **21** is based on the concept of an "entry" **27**, which contains information about some object (e.g., a person). Entries are composed of attributes **29**, which have a type and one or more values. Each attribute **29** has a particular syntax that determines what kinds of values are allowed in the attribute (e.g., ASCII characters, .jpeg file, etc.) and how these values are constrained during a particular directory operation. Entries are stored in a given table (e.g., ldap_entry) that includes entry IDs, parent IDs, a create and last modified timestamp, together with the complete entry in string format. Each attribute has an associated attribute table.

The LDAP search can be applied to a single entry (a base level search), an entry's children (a one level search), or an entire subtree (a subtree search). Thus, the scope supported by LDAP search are: base, one level and subtree. LDAP does not support search for arbitrary tree levels and path enumeration.

10 The LDAP search can be applied to a single entry (a
base level search), an entry's children (a one level
search), or an entire subtree (a subtree search). Thus, the
scope supported by LDAP search are: base, one level and
subtree. LDAP does not support search for arbitrary tree
15 levels and path enumeration.

LDAP includes an application programming interface (API), as described in "The C LDAP Application Program Interface", IETF Working Draft, July 29, 1997, which is incorporated herein by reference. An application on a given

5 client machine uses the LDAP API to effect a directory service "session" according to the flowchart of **Figure 3**. At step **40**, an LDAP session with a default LDAP server is initialized. At step **42**, an API function `ldap_init()` returns a handle to the client, and this handle may allow

10 multiple connections to be open at one time. At step **44**, the client authenticates to the LDAP server using, for example, an API `ldap_bind()` function. At step **46**, one or more LDAP operations are performed. For example, the API function `ldap_search()` may be used to perform a given

15 directory search. At step **48**, the LDAP server returns the results of the directory search, e.g., one or more database elements that meet the search criteria. The session is then closed at step **50** with the API `ldap_unbind()` function then being used to close the connection.

It may be desirable to store LDAP directory data in a backing store. **Figures 4A-4B** illustrates several representative LDAP directory service implementations that use a relational database management system (RDBMS) for this purpose. These systems merely illustrate possible LDAP directory services in which the present invention may be implemented. One of ordinary skill should appreciate, however, that the invention is not limited to an LDAP directory service provided with a DB/2 backing store. The principles of the present invention may be practiced in other types of directory services (e.g., X.500) and using other relational database management systems (e.g., Oracle, Sybase, Informix, and the like) as the backing store.

In **Figure 4A**, an LDAP client **34** can connect to a number of networked databases **38a-58n** through an LDAP server **36**. The databases **38a-38n** contain the directory information. However, from the user's perspective, the LDAP server **36** stores all the information without knowing the database **38** in which the data is actually located. With this configuration, the LDAP server **36** is freed from managing the physical data storage and is able to retrieve information from multiple database servers **38** which work together to form a huge data storage.

Figure 4B illustrates a multiple client/multiple server LDAP/DB2 enterprise solution. In this environment, a DB/2 client preferably runs on each LDAP server **36**. Each such DB/2 client can connect to any database server **38** containing
5 directory information. The collection of database servers **38a-38n** form a single directory system image, and one or more of the LDAP servers **36** can access such information. Because all the LDAP servers **36** see the same directory image, a network dispatcher **37** may be deployed to route
10 requests among the LDAP servers **36**.

One of ordinary skill should appreciate that the system architectures illustrated in **Figures 4A-4B** are not to be taken as limiting the present invention. The inventive technique may be used to search any relational database
15 using hierarchical, filter-based database queries. The present invention is a method for deferred deletion of entries in a directory service backing store. In LDAP, an entry is deleted using an SQL statement. In the prior art, the directory server responds to the delete entry statement
20 by instituting a global lock on the database tables to ensure that data in those tables cannot be modified while the entry is being deleted from the directory. The present invention provides an enhanced delete operation whereby the entry is simply marked for deletion at a later time. This
25 deferred entry deletion routine is now described in more detail below.

By way of brief background, **Figure 5** illustrates a representative portion of the entry table **60** (e.g., ldap_entry). This table comprises a number of columns including the entry identifier (EID) **62**, the parent identifier (PEID) **64**, the owner **66**, the creation date **68**, the last modified date **70**, the actual entry data **72** (in a string format), the entry size **74**, and additional information **76**. In this example, each of the entries has a given creation date. According to the invention, when a user requests deletion of an entry, the creation date for that entry is set to a given null value. **Figure 6** illustrates this operation after the user has requested deletion of entry 4. As can be seen, the null value is present in the creation data column **68**, and the timestamp for the last modified date **70** is updated to reflect when the delete request was received. At this point, the entry is conditionally deleted, even though the data for this entry remains in the entry table (as well as in the attribute tables).

Thus, according to the invention, when an entry is to be deleted, its entry in an entry table is tagged as deleted, preferably by setting its creation time to a given value (e.g., a null value). As illustrated in **Figures 5-6**,
5 this operation involves a change to only a single, unindexed field in a single row in a single table and, as a result, is quite efficient. At periodic intervals, a cleanup thread is then used to perform actual row deletions for any entry tagged as deleted. This operation is illustrated in the
10 flowchart of **Figure 7**. The thread begins at step **82** to test whether a given time period has expired. If not, the thread cycles. If, however, the time period has expired, the thread continues at step **84** to start the cleanup thread. At step **86**, the thread searches the LDAP table to identify
15 records marked for deletion. The routine then continues at step **88** to test whether all entries have been processed. If so, the routine routines to step **82**. If, however, the routine has not processed all marked entries, the routine continues at step **90** to get the next marked entry. At step
20 **92**, the routine deletes the record from the entry table. At step **94**, the routine deletes all associated records in the attribute tables. Control then returns back to step **88** and the process repeats as needed.

Because entries are merely marked for deletion, the
2 present invention also includes a routine for processing
directory search queries into the modified entry table.
Thus, as compared to the prior art, the present invention
enables a user to perform search and other queries into the
directory despite the existence of the entries tagged for
7 deletion. This operation preferably is achieved by
modifying the SQL statements to exclude rows with a null
change time, thus preventing deleted entries from being
returned by the search. This operation is illustrated
below.

By way of brief background, the following is a
representative SQL query searching for sn = Bachmann:

```
SELECT distinct EDIR.LDAP_entry.EID from  
EDIR.LDAP_entry, EDIR.LDAP-DESC where  
39 (EDIR.LDAP_entry.EID = EDIR.LDAP_DESC.DEID and  
EDIR.LDAP_DESC.AEID = 6142) and EDIR.LDAP_entry.EID in  
(select EID from EDIR.SN where SN = BACHMANN)).
```

According to the present invention, the above query
is modified to ignore entries marked as deleted,
44 preferably as follows:

```
SELECT distinct EDIR.LDAP_entry.EID from  
EDIR.LDAP_entry, EDIR.LDAP_DESC, where  
(EDIR.LDAP_entry.EID = EDIR.LDAP_DESC.DEID and  
EDIR.LDAP_DESC.AEID = 6142) and EDIR.LDAP_entry.EID in  
49 (Select EID from EDIR.SN where SN = Bachmann) and  
EDIR.LDAP_entry.Create.Timestamp <> Ø).
```

As can be seen, the last clause of the SQL statement looks for entries that have their creation timestamp as non-zero. This operation prevents deleted entries from being returned by the search.

5 As previously described, at periodic intervals, the routine tests to determine which records have been marked for deletion. This was step 76 in **Figure 7**. A representative SQL query to find records marked for deletion is then as follows:

10 Select distinct EDIR.LDAP_entry EID from
EDIR.LDAP_entry, where Create_Timestamp = Ø.

Thus, according to a preferred embodiment, a method for deleting entries from a directory in which directory information is stored in a set of database tables begins
15 upon a request to delete a directory entry. In response, the directory entry is tagged, preferably by setting the entry's creation time to a null value. If a search query is received thereafter, the method excludes tagged entries from search results that would otherwise satisfy
20 the search query. At a periodic interval, the routine then periodically searches for tagged entries, and references to the tagged entries are then deleted throughout the set of database tables. Thus, the inventive method defers entry deletions to enable
25 directory queries to be processed even if deleted entries have not yet been fully expunged from the directory.

The inventive scheme preferably takes advantage of several LDAP table structures that are now described below.

Entry table:

5 This table holds the information about a LDAP entry. This table is used for obtaining the EID of the entry and supporting LDAP_SCOPE_ONELEVEL and LDAP_SCOPE_BASE search scope.

-EID. The unique identifier of the LDAP entry. This
10 field is indexed.

-PEID. The unique identifier of a parent LDAP entry in the naming hierarchy.

-EntryData. Entries are stored using a simple text
format of the form attribute: value. Non-ASCII values or
15 values that are too long to fit on a reasonable sized line are represented using a base 64 encoding. Giving an ID, the corresponding entry can be returned with a single SELECT statement.

Descendant table:

20 The purpose of this table is to support the subtree search feature of LDAP. For each LDAP entry with an unique ID (AEID), this table contains the descendant entries unique identifiers (DEID). The columns in this table are:

25 -AEID. The unique identifier of the ancestor LDAP entry. This entry is indexed.

-DEID. The unique identifier of the descend LDAP entry.

This entry is indexed.

Attribute table:

One table per searchable attribute. Each LDAP entry is
5 assigned an unique identifier (EID) by the backing store.
The columns for this table are:

-EID

-Attribute value

Thus, in the parent table, the EID field is the unique
10 identifier of an entry in the LDAP naming hierarchy. The
PEID field is the unique identifier of the parent entry
in the naming hierarchy. In the descendant table, the
AEID field is the unique identifier of a ancestor LDAP
entry in the LDAP naming hierarchy. The DEID field is the
15 unique identifier of the descend LDAP entry.

In addition to the table structures described above, the
following SQL SELECT statements are used by LDAP/DB2
search routines:

Base Level Search:
20 SELECT entry.EntryData,
 from ldap_entry as entry
 where entry.EID in (

select distinct ldap_entry.EID
25 from -table list-

where (ldap_entry.EID=-root dn id-)

30 -sql where expressions-)

One Level Search:

 SELECT entry.EntryData,
 from ldap_entry as entry

where distinct
ldap_entry.EID
from ldap_entry, -table
list-
5 ldap_entry as pchild, -list of tables-
where ldap_entry.EID=pchild.EID
10 AND pchild.PIED=-root dn id-
-sql where expressions-)
Subtree Search
15 SELECT entry.EntryData,
from ldap_entry as entry
where entry.EID in (
select distinct ldap_entry.EID
20 from ldap_entry, ldap_desc, -table list-
where
25 (LDAP_ENTRY.EID=ldap_desc.DEID AND
ldap_desc.AEID=-root dn id-)
ldap_entry as pchild. -table list-
30 where ldap_entry.EID=ldap_desc.EID
AND ldap_desc.AEID=%d -where expressions-).
35 In the above representation, -table list- and -where
expression- are the two null terminated strings returned
by the SQL generator. The -root dn id- is the unique
identifier of the root dn. The where clause should only
be generated if -where expression- is not the empty
40 string and no errors where detected in the parsing the
LDAP filter.

As noted above, the invention may be implemented in any hierarchical directory service in which a relational database management system (RDBMS) is used to provide a backing store function. Thus, for example, the

5 principles of the invention may be carried out in an X.500 directory service or hereinafter-developed LDAP implementations. The SQL query generated according to the present invention is used to access the relational database, and results are then returned in response to

10 this query. The invention may also be implemented within a relational database management system being used as an add-on to a directory service. One of ordinary skill will appreciate that the invention can be applied to any relational database management system (RDBMS) and not

15 simply DB/2, the implementation described above. Thus, for example, the relational database may be Oracle, Sybase or any other third party supplied backing store. In addition, the EID sets approach can also be applied to b-tree based LDAP server implementation.

1 Moreover, although the preferred embodiment has been
described in the context of deleting an LDAP entry in a
relational database backing store, the inventive
technique should be broadly construed to extend to
deleting entries from any type of directory in which
6 directory information is stored in a set of database
tables. Thus, the present invention is not limited to
use with hierarchical directories. Rather, as noted
above, the techniques described herein may be implemented
in conjunction with any higher level directory structure
11 in which information is spread out over a set of tables.

One of the preferred embodiments of the routine of
this invention is as a set of instructions (e.g.,
computer program code) in a code module resident in or
35 downloadable to the random access memory of a computer.
Until required by the computer, the set of instructions
may be stored in another computer memory, for example, in
a hard disk drive, or in a removable memory such as an
optical disk (for eventual use in a CD ROM) or floppy
40 disk (for eventual use in a floppy disk drive), or
downloaded via the Internet or other computer network.

In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would

5 also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is set

10 forth in the following claims.

CLAIMS

1. A method for deleting entries from a directory in which directory information is stored in a
15 set of database tables, comprising the steps of:

responsive to a request to delete a directory entry, tagging the directory entry in a first table;
periodically searching for tagged entries in the first table during a cleanup process interval; and
deleting references to the tagged entries throughout the set of database tables.

2. The method as described in Claim 1 wherein
24 the directory entry is tagged by setting its creation time to a given value.

3. The method as described in Claim 2 wherein
the given value is a null value.
29

4. The method as described in Claim 1, further including the steps of:
performing a search for directory entries that satisfy a search query; and
34 excluding tagged entries from search results that otherwise satisfy the search query.

5. The method as described in Claim 4 wherein the step of excluding tagged entries includes modifying an SQL query to exclude rows having a null change creation.

5

6. The method as described in Claim 1 wherein the directory is a Lightweight Directory Access Protocol (LDAP) directory service and the database tables are managed by a relational database management
10 service.

7. The method as described in Claim 1 wherein the first table is an entry table.

15 8. The method as described in Claim 7 wherein the set of database tables includes at least one attribute table storing information about an attribute.

2 9. A method for deleting entries from a
directory in which directory information is stored in a
set of database tables, comprising the steps of:
responsive to a request to delete a directory
entry, tagging the directory entry in a first table;
7 responsive to a search for directory entries that
satisfy a search query, excluding tagged entries from
search results that otherwise satisfy the search query;
periodically searching for tagged entries during a
12 cleanup process interval; and
deleting references to the tagged entries
throughout the set of database tables.

10. The method as described in Claim 9 wherein
35 the directory entry is tagged by setting its creation
time to a given value.

11. The method as described in Claim 10 wherein
the given value is a null value.

40

12. The method as described in Claim 9 wherein
the first table is an entry table.

13. The method as described in Claim 12 wherein the set of database tables includes at least one attribute table storing information about an attribute.

5 14. A method for searching a database from a directory service, comprising the steps of:

responsive to a search for directory entries that satisfy a search query, excluding given entries from search results that otherwise satisfy the search query,
10 wherein the given entries identify database entries that have been tagged for deletion; and
returning the search results.

15 15. The method as described in Claim 14 wherein the directory service is a Lightweight Directory Access Protocol (LDAP) directory service and the database is managed by a relational database management service.

16. A computer program product in a
20 computer-readable medium for deleting entries from a directory in which directory information is stored in a set of database tables, comprising:

means responsive to a request to delete a directory entry for tagging the directory entry in a
25 first table;

means for periodically searching for tagged entries in the first table during a cleanup process interval; and

means for deleting references to the tagged entries throughout the set of database tables.

17. The computer program product as described in
33 Claim 16, further including:

means responsive to a search for directory entries that satisfy a search query for excluding tagged entries from search results that otherwise satisfy the search query.

38

18. The computer program product as described in Claim 17 wherein the search query is a Lightweight Directory Access Protocol (LDAP) directory service query.

43

19. A directory service, comprising:

a directory organized as a naming hierarchy having a plurality of entries each represented by a unique identifier;

48 a relational database management system having a backing store for storing directory data in a set of database tables; and

means for deleting entries from the directory,
comprising:

3 means responsive to a request to delete a
directory entry for tagging the directory entry in
a first table;

 means for periodically searching for tagged
entries in the first table during a cleanup
8 process interval;

 means for deleting references to the tagged
entries throughout the set of database tables; and

 means responsive to a search for directory
entries that satisfy a search query for excluding
35 tagged entries from search results that otherwise
satisfy the search query.

20. The directory service as described in Claim
19 wherein the directory is compliant with the
40 Lightweight Directory Access Protocol (LDAP).

METHOD FOR DEFERRED DELETION OF ENTRIES
FOR A DIRECTORY SERVICE BACKING STORE

3

ABSTRACT OF THE DISCLOSURE

A method for deleting entries from a directory in which directory information is stored in a set of database tables begins upon a request to delete a directory entry. In response, the directory entry is tagged, preferably by setting the entry's creation time to a null value. If a search query is received thereafter, the routine excludes tagged entries from search results that would otherwise satisfy the search query. Periodically, the routine searches for tagged entries, and references to the tagged entries are then deleted throughout the set of database tables. Thus, the inventive method defers entry deletions to enable directory queries to be processed even if deleted entries have not yet been fully expunged from the directory.

1/4
AT9-98-955
BACHMANN, D. W. et al.

FIG. 1

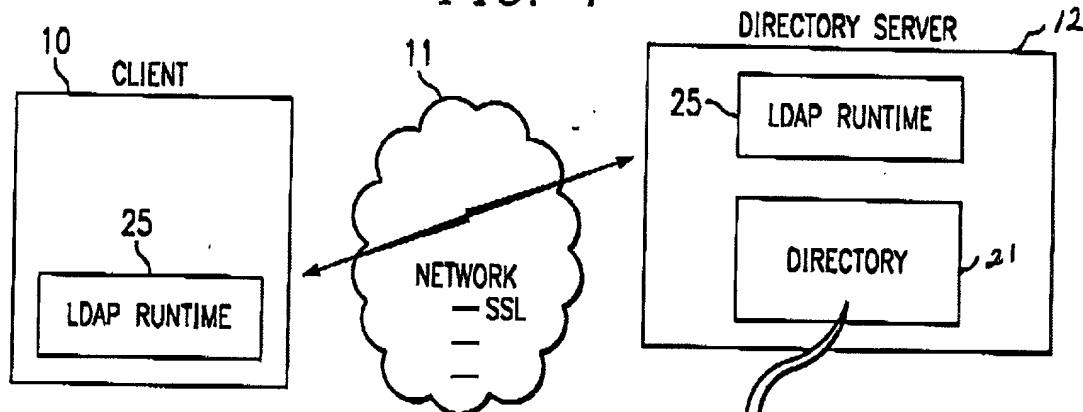


FIG. 2

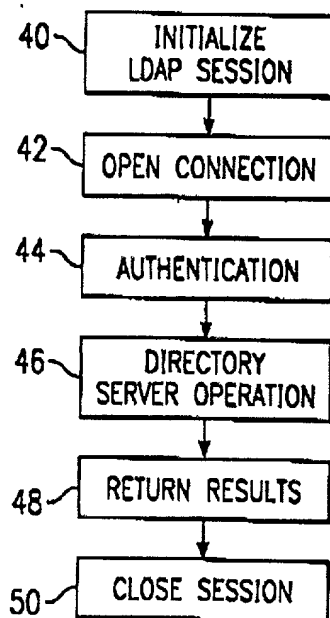
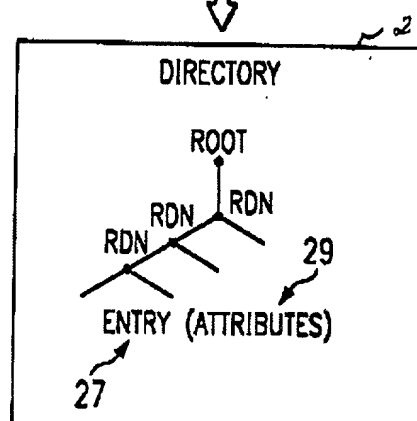


FIG. 3

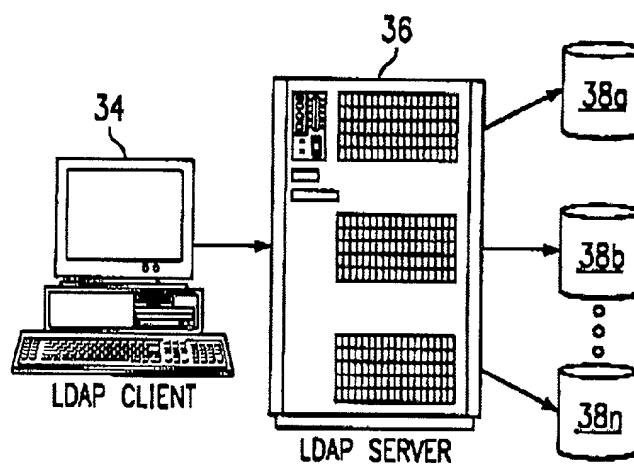


FIG. 4A

2/4

AT9-98-955

BACHMANN, D. W. et al.

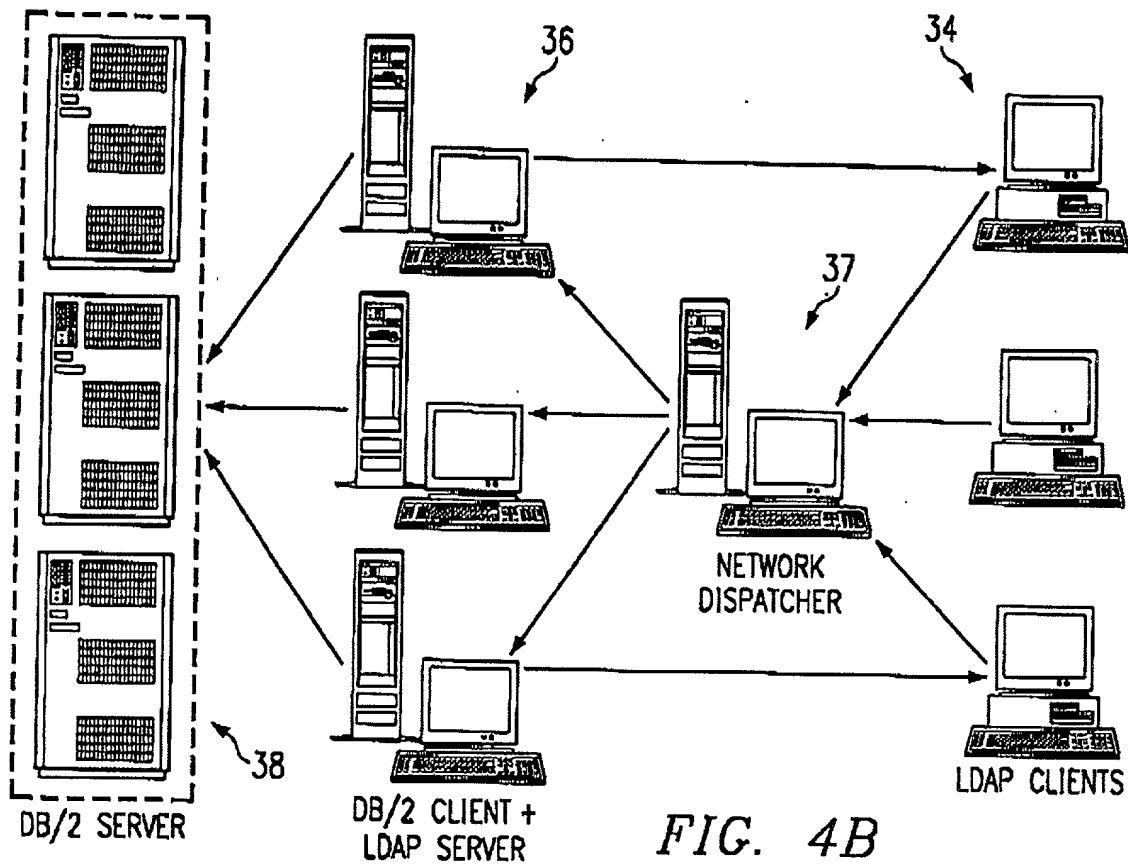


FIG. 4B

66340 336260

AT9-98-955
BACHMANN, D. W. et al.

| EID | PEID | OWNER | creation | modify | entry data | SIZE | other |
|-----|------|-----------------------|--------------------|--------------------|---------------------------------|------|-------|
| 3 | 1 | on = IBM-US c = US | 1:01 am 11/7/97 | 6:30 am 3/22/99 | "dn: on = Austin..." | 511 | Ø |
| 4 | 3 | on = IBM-US c = US | 5:12 am 4/3/98 | 7:10 am 3/22/99 | "dn: on = Dave Bachmann ..." | 632 | Ø |

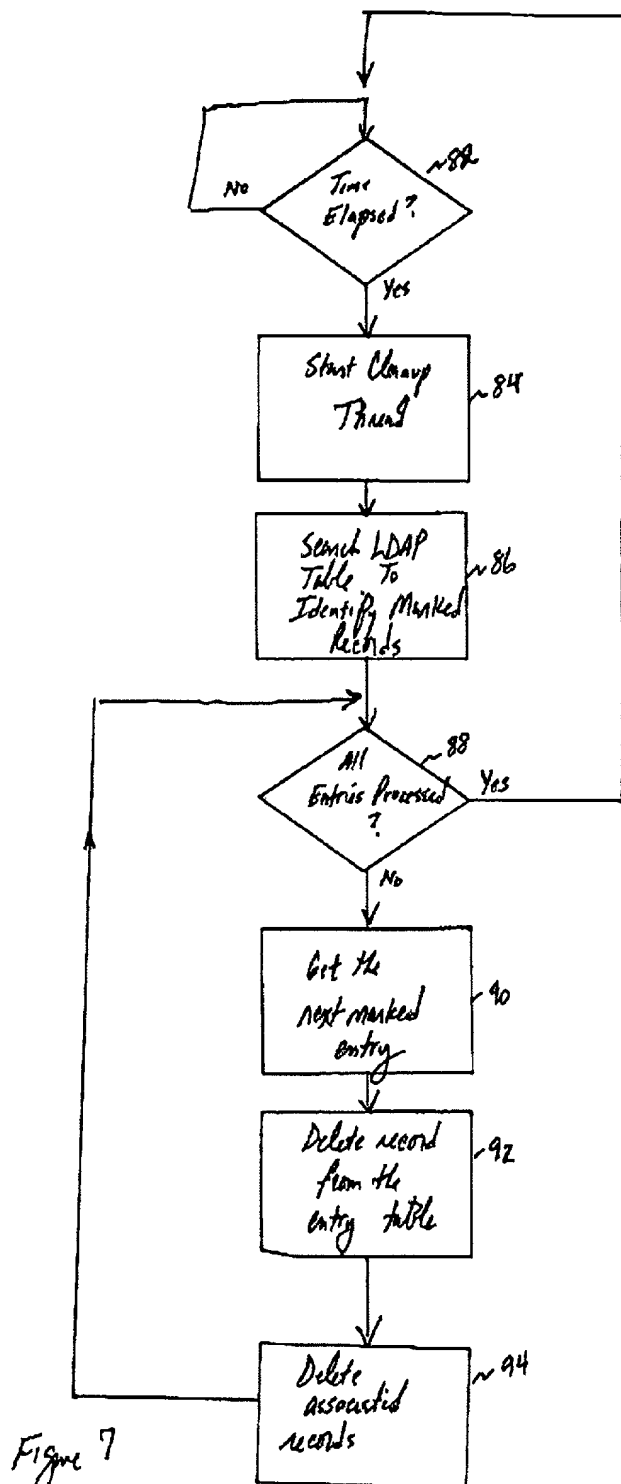
60 ↗

Figure 5

| EID | PEID | owner | creation | modified | entry data | size | status |
|-----|------|------------------|-------------------|--------------------|--------------------|------|--------|
| 3 | 1 | o=ibm_us | 4:01 am 1/7/97 | 6:30 am 3/22/99 | "dn: cn=Austin..." | 511 | φ |
| 4 | 3 | o=ibm_us c=us | φ | 8:35 am 3/22/99 | "dn: cn=Dave..." | 652 | φ |

Figure 6

4/4
AT9-98-955
BACHMANN, D. W. et al.



DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD FOR DEFERRED DELETION OF ENTRIES FOR A DIRECTORY SERVICE BACKING STORE

the specification of which (check one):

: is attached hereto.

9 was filed on _____;
as Application Serial No. _____
and which was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Priority Claimed

| | | | | |
|----------|-----------|------------------|---------|--------|
| (Number) | (Country) | (Day/Month/Year) | ___ Yes | ___ No |
|----------|-----------|------------------|---------|--------|

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, § 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

| (Application Serial #) | (Filing Date) | (Status) |
|------------------------|---------------|----------|
|------------------------|---------------|----------|

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; James H. Barksdale, Jr., Reg. No. 24,091; Thomas E. Tyson, Reg. No. 28,543; Robert M. Carwell, Reg. No. 28,499; Jeffrey S. LaBaw, Reg. No. 31,633; Douglas H. Lefevre, Reg. No. 26,193; Casimer K. Salys, Reg. No. 28,900; David A. Mims, Jr., Reg. No. 32,708; Anthony V. England, Reg. No. 35,129; Vollel Emile, Reg. No. 39,969; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; David H. Judson, Reg. No. 30,467, and Douglas A. Sorensen, Reg. No. 31,570.

Send correspondence to: David H. Judson, Hughes & Luce, L.L.P., 1717 Main Street, Suite 2800, Dallas, Texas 75201 and direct all telephone calls to Mr. Judson at 214/939-5672.

FULL NAME OF FIRST
INVENTOR
INVENTOR'S SIGNATURE:

David W. Bachmann

David W. Bachmann

DATE:

4/28/1999

RESIDENCE:

15105 Harkey Cove
Leander, Texas 78641

CITIZENSHIP:

US

FULL NAME OF FIRST
INVENTOR
INVENTOR'S SIGNATURE:

Terry L. Dunkle

Terry L. Dunkle

DATE:

4/28/99

RESIDENCE:

1102 Oakwood Blvd.
Round Rock, Texas 78681

CITIZENSHIP:

US

006372.00203:0428554.01